SOURCECODE

```
#include<stdio.h>
#include<stdlib.h>
double Median(int* a, int n1, int* b, int n2)
 {    // if a is larger we conduct search on b:
    if (n1 > n2) return Median(b, n2, a, n1);
    int n = n1 + n2;
    int left = (n1 + n2 + 1) / 2; // length of left half of virtual sorted
array
    // apply binary search on smaller array
    int low = 0, high = n1;
    while (low <= high) {
        int mid1 = (low + high) /2;//how many to take from first array to form
left half of virtually combined array
        int mid2 = left - mid1;//how many remaining to take froms second array
to complete left half of virtually combined array
        // calculate l1, l2, r1, and r2;
        int left1 = INT_MIN, left2 = INT_MIN,right1 = INT_MAX, right2 =
INT_MAX;
        //ensuring no index out of bounce
        if (mid1 < n1)
             right1 = a[mid1];
        if (mid2 < n2)
             right2 = b[mid2];
        if (mid1 - 1 >= 0)
             left1 = a[mid1 - 1];
        if (mid2 - 1 >= 0)
            left2 = b[mid2 - 1];
        if (left1 <= right2 && left2 <= right1)//point of symmetry in
virtually combiend array located
            {
               if (n % 2 !=0) //odd number of elements
                 {   if (left1 > left2)
                         return (double)left1;
                     else
                         return (double)left2;
                 }
                 else {//even number of elements
                     if (left1 > left2)
                         return ((double)left1 + (double)(right1 < right2 ?
right1 : right2)) / 2.0;
                     else
                         return ((double)left2 + (double)(right1 < right2 ?
right1 : right2)) / 2.0;
                 }
        }
        else if (left1 > right2)
      high = mid1 - 1;
        else
      low = mid1 + 1;
    }
    return 0;
}
int main()
 {   int n1,n2;
      printf("enter the size of first array-\n");
      scanf("%d",&n1);
      printf("enter the size of second array-\n");
      scanf("%d",&n2);

      int *a=(int*)malloc(n1*sizeof(int));
      int *b=(int*)malloc(n2*sizeof(int));
      int i,j;
      printf("enter first array-\n");
      for(i=0;i<n1;i++)
       scanf("%d",&a[i]);
      printf("enter second array-\n");
      for(i=0;i<n2;i++)
       scanf("%d",&b[i]);
```

```
        printf("\nMedian-%f",Median(a,n1,b,n2));
}
```

OUTPUT

```
C:\Users\K GANGULY\Documents\C files\MedianofTwoSortedArrays.exe
enter the size of first array-
5
enter the size of second array-
6
enter first array-
1
3
5
7
9
enter second array-
2
4
6
8
10
12

Median-6.000000
--------------------------------
Process exited after 28.77 seconds with return value 16
Press any key to continue . . .
```

```
C:\Users\K GANGULY\Documents\C files\MedianofTwoSortedArrays.exe
enter the size of first array-
4
enter the size of second array-
5
enter first array-
12
14
19
42
enter second array-
1
13
26
50
65

Median-19.000000
--------------------------------
Process exited after 33.46 seconds with return value 17
Press any key to continue . . .
```

SOURCECODE

```
#include <stdio.h>
#include<stdlib.h>
//Count Sort
void countingSort(int arr[], int n, int exp)
{   int output[n],i;
    int dig_freq[10] = {0};
    // Count occurrences of each digit in the current place value
    for (i = 0; i < n; i++) {
        dig_freq[(arr[i] / exp) % 10]++;
    } // store cumulative count
    for (i = 1; i < 10; i++)
{    dig_freq[i] += dig_freq[i - 1];
    } // Build the output array using count array
    for (i = n - 1; i >= 0; i--)
 {      output[dig_freq[(arr[i] / exp) % 10] - 1] = arr[i];
        dig_freq[(arr[i] / exp) % 10]--;
    } // Copy the output array to the original array
    for (i = 0; i < n; i++)
  {        arr[i] = output[i];

}
//Radix Sort
void RadixSort(int *a,int n)
{     //finding max
    int max=a[0];
      int i,exp;
      for(i=0;i<n;i++)
       max=a[i]>max?a[i]:max;

    for(exp=1;max/exp>0;exp *=10)
        countingSort(a,n,exp);
}int main() {
    int n1;
    printf("Enter the size of the array: ");
    scanf("%d", &n1);
    int *a = (int *)malloc(n1 * sizeof(int));
    int i;
    printf("Enter the array:\n");
    for (i = 0; i < n1; i++)
        scanf("%d", &a[i]);
    RadixSort(a, n1);
    printf("\nSorted array:\n");
    for (i = 0; i < n1; i++)
        printf("%d ", a[i]);
    free(a);
    return 0;
}
```

OUTPUT

```
C:\Users\K GANGULY\Documents\C files\RadixSort.exe
Enter the size of the array: 5
Enter the array:
3
4
1
2
5

Sorted array:
1 2 3 4 5
--------------------------------
Process exited after 9.169 seconds with ret
Press any key to continue . . .
```

```
C:\Users\K GANGULY\Documents\C files\RadixSort.exe
Enter the size of the array: 6
Enter the array:
12
1
53
78
4
2

Sorted array:
1 2 4 12 53 78
--------------------------------
Process exited after 12.49 seconds with re
Press any key to continue . . .
```